2008 10th Intl. Conf. on Control, Automation, Robotics and Vision
Hanoi, Vietnam, 17-20 Dec 2008
pp. 1253-1258

Robot Path Planning in Dynamic Environments Using a Simulated Annealing Based Approach

Hui Miao and Yu-Chu Tian
Faculty of Information Technology
Queensland University of Technology
Box 2434, Brisbane QLD 4001, Australia
y.tian@qut.edu.au

***Abstract*—This paper proposes a simulated annealing based approach to determine the optimal or near-optimal path quickly for a mobile robot in dynamic environments with static and dynamic obstacles. The approach uses vertices of the obstacles to define the search space. It processes off-line computation based on known static obstacles, and re-computes the route online if a moving obstacle is detected. The contributions of the work include the employment of the simulated annealing algorithm for robot path planning in dynamic environments, and the development of a new algorithm planner for enhancement of the efficiency of the path planning algorithm. The effectiveness of the proposed approach is demonstrated through simulations under typical dynamic environments and comparisons with existing methods.**

***Keywords*—simulated annealing algorithm, robot path planning, dynamic environement**

I. Introduction

Path planning is one of the most important aspects in automatic robot navigation. Generally, depending on how much the robot knows about the environment around, two types of path planning have been investigated:

1) Path planning based on clearly known environments, i.e., the robot has already known the location of the obstacles before starting to move. The path of the robot could be computed off-line, and the globally optimal path can be obtained because the entire environment is known.
2) Path planning based on partly known or uncertain environments. The robot probes the environment by using sensors to acquire the information about the location, shape and size of the obstacles, and uses the local information to perform online path planning.

Nowadays, the techniques for robot path planning in static and known environments are relatively mature. Methods and algorithms for this type of path planning have been well developed with hundreds of publications in the open literature. Representative methods are Visibility Graph [1], Voronoi diagrams [2], and Grids [3]. Given the entire information of the environment, the globally optimal or near-optimal path could be found using some algorithms such as the Genetic Algorithm [4]. Davidor [5] developed a tailored genetic algorithm with a modified crossover operator to optimize the robot path. Nearchou [6] used the number of vertices produced in visibility graphs to build fixed length chromosomes in which the presence of a vertex within the path is indicated by setting a bit at the appropriate locus. This method employed a reordering operator to enhance the performance, and was capable of determining a near-optimal solution. Cai and Peng [7] developed a fixed-length decimal encoding mechanism to replace the variable-length encoding methods and other fixed-length binary encoding approaches in genetic approach for robot path planning.

However, as described in [8] and [9], quite often, a robot could not predict the environment around because the status and movement of the obstacles change all the time. The robot cannot make a one time global path planning in the environment. It has to acquire the surrounding environment information using sensors and then to perform online and real-time path planning. As a result, complexity and uncertainty of robot path planning increase significantly in dynamic environments. Therefore, traditional path planning algorithms, such as Visibility Graph, Voronoi diagrams, and Grids, do not perform well in dynamic environments. How to manipulate the robot in dynamic environments to travel to the destination safely and optimally without collision is a major issue to be addressed.

Limited reports in the open literature have discussed the optimal path planning for a robot in dynamic environments. Lv and Feng [10] introduced numerical potential field to find the path for the robot in dynamic environments. They used an ant colony optimization algorithm to perform optimal path searching. Xu, Xie, and Xie [11] also investigated dynamic path planning using the concepts of artificial potential field and genetic algorithm. Recently, Wang, Sillitoe and Mulvaney [12] introduced a genetic algorithm planner to determine optimal or near-optimal solutions for a mobile robot in dynamic environments.

Implementing path planning algorithms in a real robot in dynamic environments is also limited. Cao, Huang and Zhou [13] proposed an evolutionary artificial potential field algorithm for dynamic path planning for soccer robots in RoboCup 2005. The method was used to plan the path for mobile robots in a dynamic environment where the target and obstacles are moving. In [13], the new force function and relative threat coefficient function were defined first. Then, a new potential field path-planning algorithm based on the relative threat coefficient was presented. Finally, computer simulation and experiment were conducted to demonstrate the effectiveness of the dynamic path planning scheme.

However, the methods mentioned above all have some drawbacks:

- When dealing with robot path planning in complex environments, the genetic algorithm based approaches are computationally intensive and time consuming. According to the simulation results in [12], the genetic algorithm requires near 30 seconds to find the first feasible path for the robot in the environment with 14 static obstacles and 5 dynamic obstacles.
- The other two methods [14] and [13] ignored the dimension of the obstacles. They considered an obstacle as a point or simple square block which could generate the repulsive force to robots. Therefore, the methods are not suitable for optimal path searching in a variety of environments with sharp obstacles. For searching the optimal path for a robot, dimensions of the obstacles should not be ignored.

Aiming to overcome those drawbacks, this work employs the simulation annealing algorithm [15] for robot path planning in dynamic environments. The simulated annealing based approach could determine the optimal or near-optimal robot path efficiently through various shapes of static and dynamic obstacles. The obstacles are all described as polygons; this is more realistic than the assumption made in [14] and [13]. The simulated annealing algorithm is a generic and probabilistic meta-algorithm for global optimization problems. It is able to locate a good approximation to the global optimum of a given function in a large search space. Simulated annealing based approaches have already been used in searching optimal path in stationary path planning methods in [16] and [14]. However, they have not been effectively applied to dynamic robot path planning. This motivates the work of this paper.

II. Approach Description

This section develops the simulated annealing algorithm based approach for dynamic robot path planning. We will discuss in detail the modelling of the environment, structure of the approach/algorithm, the generation of the initial feasible path, the new planner for generating the random path, and the procedure of the online computation.

A. Environment Modelling and Assumptions

In our approach, both moving obstacles and static obstacles are represented as bounding polygons. The vertices of the obstacles in the environment form the search space for the algorithm. The dynamic environment is designed to contain stationary and moving obstacles. The trajectory of a moving obstacle is constituted by a series of polygons with their positions being updated along with the time. Any motion parameters, such as speed and direction, of the dynamic obstacle can be made available to the robot when the obstacle is in the range of the sensor. The robot could change its moving direction at any time.

The procedure of the proposed path planning algorithm in dynamic environments is divided into two stages:

Step 1: Off-line path computation based on the information of the stationary obstacles; and

Step 2: Online path computation once the moving obstacles are detected by the sensor of the robot.

In offline computation, the entire information about the location of the vertices of the static obstacles is known to the robot. The algorithm begins with the computation of the optimised path for the robot based on the positions of the stationary obstacles before starting to travel. Once the computation is complete, the robot can start to move through the stationary obstacles.

The movement and the trajectory of a moving obstacle in dynamic environments are unknown to the robot. The robot will sense the moving information of the dynamic obstacle. It is assumed that the function of the robot sensor is to acquire the moving parameter of the dynamic obstacle, and the sensor can detect 360-degree direction of the robot. When a moving obstacle enters the detection range of the robot, the sensor will detect the obstacle and the robot acquires the moving information, such as speed and moving direction, of the obstacle. Then, the robot uses the motion information to compute the possibility of clashing of the robot with the moving obstacle. If the moving obstacle will not hit the robot, the robot will use the current path plan to travel through the map. To make the algorithm practically feasible, the processing time for the robot to re-compute a new feasible path when a moving obstacle is detected is a key performance metric and has to be sufficiently short.

For realistic simulations, as in [17], all obstacles in the map are enlarged by a fixed value so that the robot could approach obstacles without collision. The dimension of the robot is neglected, and consequently the robot is regarded as a single point. In Figure 1, the black polygons represent static obstacles and the hollow polygons are moving obstacles. All the obstacles are enlarged by some values, i.e., additional margins are created, to prevent possible collision of the robot with the obstacles. The vertices of the enlarged polygons form the search space for the robot.

A mathematic model of determining the possibility of robot colliding with a moving obstacle could be developed. The model is described as follows:

1) The first crossing point between the robot path proposed by the planner and the trajectory of a moving obstacle is computed before examining the possibility of collision;
2) Based on the time t required for the robot to cover the distance from the current position to the first crossing point, the instantaneous location of the moving obstacle and consequently the exclusion area for this obstacle can be computed;
3) If the robot path between the vertex and the crossing point across the edges of the moving obstacle in odd times, then a collision would occur between the robot and the moving obstacle, otherwise the collision will not happen.

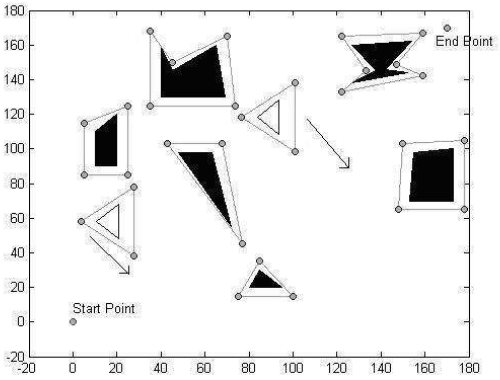

Fig. 1. Example of the dynamic environment.

B. Algorithm Structure and Expressions

Traditionally, the path length E_f is the evaluation criterion for the quality of the path solution derived from the algorithm. The shorter the path, the better the solution is.

A feasible path solution is expressed by a series of vertices linking the start point through to the end point. Each vertex of the obstacle has its series number; and a path is represented by a sequence of vertex numbers. Therefore, a feasible path solution X is described as:

$$X = \{V_{start}, V_{start+1}, V_{start+2}, \cdots, V_{end-1}, V_{end}\} \quad (1)$$

The evaluation function E_f is given by:

$$E_f = \sum_{i=start}^{i=end-1} D(V_i, V_{i+1}) \quad (2)$$

where $D(V_i, V_{i+1})$ represents the direct distance from vertex V_i to V_{i+1}. The pseudo-code of the algorithm is as following:

$T = T_{initial}$;
while $(T > T_{terminate})$
  randomly generate one feasible solution X_s;
  evaluate X_s, $E_f = f(X_s)$;
  $count = 1$;
  while $(count < Threshold)$
    generate a new feasible solution X_n *base on* X_s;
    evaluate X_n, $E_n = f(X_n)$;
    if $f(X_n) < f(X_s)$
      $X_s = X_n$;
    else if $rand(1) < (exp((f(X_s) - f(X_n))/T)$
      $X_s = X_n$;
      $count = count + 1$;
    endif
  endwhile
  $T = cool\_rate * T$;
  update X_s *at each reduction of temperature* T
endwhile
X_s *is the optimal or near-optimal solution for the robot.*

C. Initial Path Selection Process

It is known from Subsection B above that after each reduction of initial temperature T, a new feasible solution X_n is selected in each new round. It is essential for the algorithm to quickly and correctly generate a random feasible path in each round for the robot. For this purpose, the edges of the static obstacles should be specified first. An edge is any strait-line between two points on the edge of or within the obstacles. Any path section crosses the edge is defined as an invalid path. In the proposed program, a separate array is used to store all the edges of the map.

In the initial stage of the program, except dynamic objects, starting point, and end point, a vertex is chosen randomly from the map. Then, use a strait line to connect the start point and the selected vertex. If the path line intersects with any edges of the obstacle in the map, the path is recognized as an invalid path. Another vertex will be randomly selected again for testing. If the strait line to the selected vertex does not intersect with any edges in the map, the strait line is recognized as part of a valid path. Then add the vertex into the path. After that, start from the selected vertex to find next valid vertex. Keep doing the above procedure until the end point is selected and also the path line to the end point is a valid path, i.e., the path to the end point does not intersects with any edges.

D. Random Path Planner

Different from the simple path planner that was previously used in [10], a more complex random path planner is developed in this paper. Additional deleting and switching operators are used in the planner to generate a new solution by flipping some bits of the X_s. The algorithm randomly chooses one operator to generate the new path. As shown in Figure 2, the deleting operator randomly deletes one vertex from the initial X_s to generate a new solution, while the switching operator randomly swaps two vertices in X_s. As the same selection criteria in generating initial path, each line section generated by the operators should be firstly tested against the edges in the map in order to generate a valid path line. This means that the line section created does not intersect with any edges in the map.

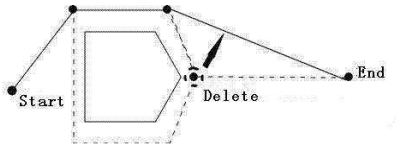

Fig. 2. Deleting operation.

Since the path length is the evaluation criterion, it is believed that randomly deleting vertices could contribute more to the reduction of the path length of the solution. Therefore, the possibility of choosing the deleting operator is set to be higher than that of selecting the switching operator. In our simulation program that will be discussed later, the possibility of choosing the deleting operator is set to be 0.78. After generating a new

path solution by the deleting or switching operator, the new solution will be evaluated using the evaluation function. Either accept the new solution if it is better than the previous one; or accept the solution in a certain probability characterised by the current annealing temperature.

E. Online Path Planning

As stated in Introduction section, while a robot uses the route generated by off-line planning to travel through static obstacles, the online path planner is triggered automatically to compute an alternative optimal path when a dynamic obstacle is detected.

The moving information of the dynamic obstacles gathered by the sensor of the robot includes speed and moving direction. With this acquired information together with the moving parameter of the robot, the robot could infer the possibility of collision with the moving obstacles.

The simulated annealing optimization algorithm for finding optimal path is triggered when the computation shows that the robot will collide with the moving obstacle if no change of movement will be made in the future. If the moving obstacle will not hit the robot, the robot will use the current path plan to travel through the map. If it is computed that a new alternative path should be created, the simulated annealing algorithm will be activated and reloaded with the updated search space. The current status of the robot, the location information of the dynamic obstacles that could cause the collision and the location information of the static obstacles are all combined as a new search space for the robot.

With the increasing number of the static and dynamic obstacles, the search space becomes larger. As a result, the time required for planning a path becomes longer. Therefore, in a complicated situation with many static and dynamic obstacles, it is required that the planning time is relatively short for robot to change the directions to avoid the collision with the obstacles.

III. Simulation Results

Simulation experiments are carried out in this work using math simulation software Matlab 7.1 [18] under Windows. The hardware configuration is Intel Pentium M 1.6GHz CPU with 256MB memory.

A. Simulation Environment and Control Parameters

The algorithm is tested for four different environments. Each environment contains static and dynamic obstacles. The path is optimized for length. The solution derived from the algorithm is optimal or near-optimal. The numbers of the static and dynamic obstacles in the four testing environments are tabulated in Table I. The numbers of the static vertices for off-line planning are also listed in Table I.

In our simulation experiments, the dynamic obstacles have random shapes in all four cases. The first two environments simulate simple scenarios where the dynamic obstacles appear simultaneously and travel simply forward in the same direction. The last two environments are more complicated scenarios where the dynamic obstacles do not appear simultaneously but each appears at a random time and moves forward or backward. Also, the numbers of the static and moving obstacles are larger than those of the simple environments. All the position information of the static obstacles in the map is known to the robot for off-line planning before the robot starts to travel.

TABLE I
FOUR TESTING ENVIRONMENTS.

Environment	No. of Static Obstacles	No. of Dynamic Obstacles	No. of Static Vertices
1	3	2	10
2	6	2	25
3	9	4	53
4	14	6	82

TABLE II
CONTROL PARAMETERS FOR SIMULATION ANNEALING.

Initial Temperature	Terminate Temperature	Cooling Rate	Deleting Operator Rate	Switching Operator Rate
999999999	555555555	0.97	78%	22%

The control parameters for simulated annealing algorithm are set in traditional way. According to [16], with bigger initial temperature and smaller cooling rate, it is much more likely to find the optimal solution, but it will require longer processing time for running. After many times of testing, we set control parameters of the algorithm as shown in Table II

B. Results and Discussions

Because the first two environments simulate simple scenarios where the dynamic obstacles appear simultaneously and travel simply forward in the same direction, the most complicated scenario, Environment 4, is presented first to illustrate the simulation results.

In Environment 4, there are 14 static obstacles and 6 dynamic obstacles. Furthermore, for creating complexity, dynamic obstacles appear randomly at different times and move in different directions. In Figure 3, the series of hollow polygons are the trajectory of the dynamic obstacles. The obstacles move along in the direction where the arrow indicates. The line constituted by a series of points is the trajectory of the robot. Another arrow is used to indicate the moving direction of the robot.

Figures 3 and 4 show that the dynamic obstacles do not appears simultaneously, four dynamic obstacles will appear at random times after first two dynamic obstacles come out. The dynamic obstacles are numbered in Figure 4. Figure 4 illustrates the simulation results of the algorithm using offline and online planning to find an optimal or near-optimal path for the robot in the simulated dynamic environment.

It is seen from Figure 4 that the robot changes its route to avoid collision with dynamic obstacles 1 and 2. Also, the new path replaces the original path and is still an optimal or near

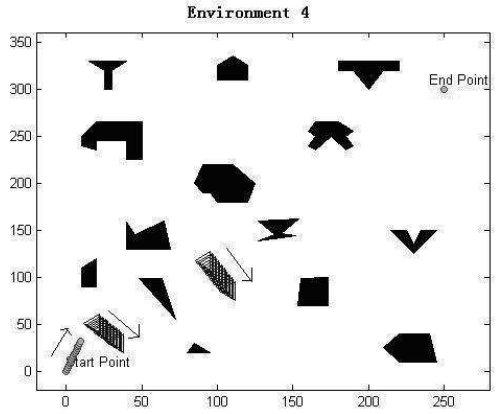

Fig. 3. Dynamic obstacles appearing randomly in environment 4.

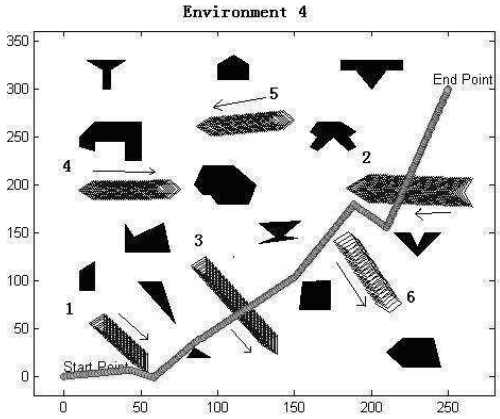

Fig. 4. Path planning in dynamic environment with 14 static objects and 6 dynamic objects.

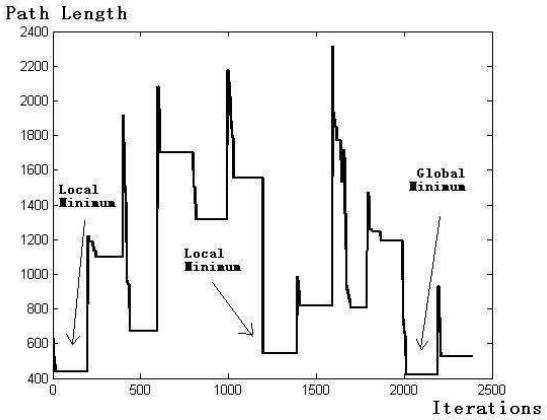

Fig. 5. Convergence of the algorithm.

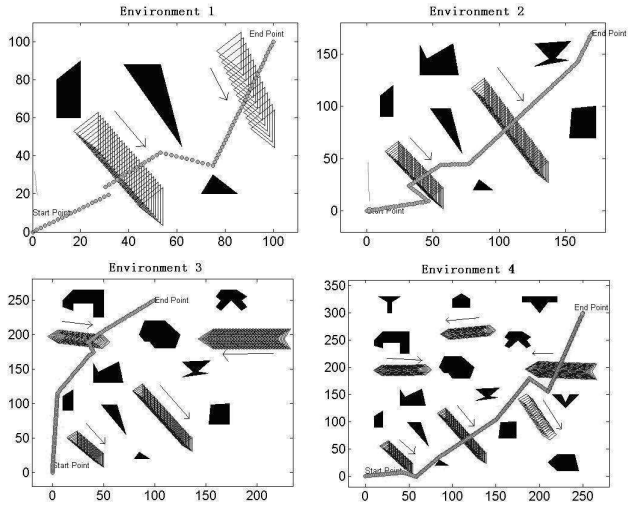

Fig. 6. Path planning in four different environments.

optimal path. It is also observed from the figure that the robot will use current path to travel if it no collision is predicted. The robot's computation shows that dynamic obstacles 5 and 6 will not affect its trajectory, so the robot uses the current path to travel passing the obstacles in the map.

Figure 5 illustrates the convergence of the algorithm. It is seen from Figure 5 that the result converges rapidly in each round of temperature reduction. The algorithm could jump out of the local minimum to converge to the global minimum.

Figure 6 depicts the simulation results of the four environments. It is seen from the plots in Figure 6 that the paths generated from the proposed algorithm are all optimal or near-optimal. The robot has the ability to use online path planning to change the route to avoid collision with the moving obstacles.

The efficiency of the proposed algorithm is also evaluated quantitatively. Compared with previous work in [17] and [12], the proposed algorithm could compute the alternative path relatively quickly for the robot to avoid collision with the moving obstacles as evidenced by the fact that the planning time of the proposed algorithm is observed to be relatively short to make a motion change.

In order to highlight the high efficiency of the algorithm, we focus on online planning. When online planning is triggered, the processing time for generating an alternative path is recorded. The online planner is tested 10 times for each environment. From the results of the 10 tests, the average value is computed as the efficiency metric for the algorithm. The same control parameters are used in all four environments.

As shown in Table III, the processing times of the proposed algorithm for all simulated environments are acceptable in comparison with previous results in [12] and [17]. The efficiency could be further improved through modifying the control parameters or implementing the algorithm in low-level languages such as C/C++.

In [17], a modified genetic algorithm based approach is introduced to optimize the path for a robot in static environments. The method also uses vertices as search space. Therefore, comparisons between the genetic algorithm based approach and the proposed simulated annealing based approach can be made through comparing the processing times of the two

TABLE III
PROCESSING TIME IN EACH OF THE SIMULATED ENVIRONMENTS.

Environment	1	2	3	4
Number of vertices	10	25	53	82
Processing Time (s)	0.57	1.201	4.784	13.57

approaches in the same search space on an identical computer.

Both two methods are tested on a personal computer equipped with Intel Pentium Duo 2 Core 1.6GHz Processor. Figure 7 shows the simulation results of the two methods: the left plot is for the genetic algorithm based approach [17], while the right plot is for the simulated annealing algorithm based approach developed in this paper.

Figure 8 compares the processing times of the two approaches for the same number of vertices. Each of the two approaches is run 10 times for the same environment. The results in the figure are the medians obtained over the 10 runs. It is seen from Figure 8 that, compared with the genetic algorithm based approach presented in [17], the proposed approach requires noticeably shorter processing time for determining the optimal robot path.

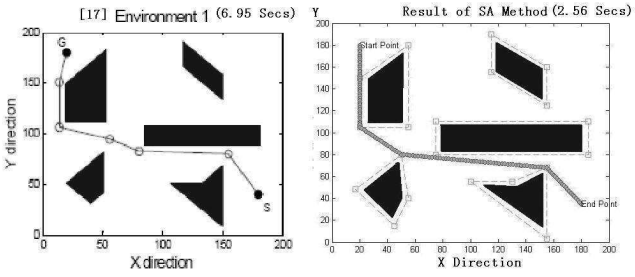

Fig. 7. Comparisons of the planned path between the two methods.

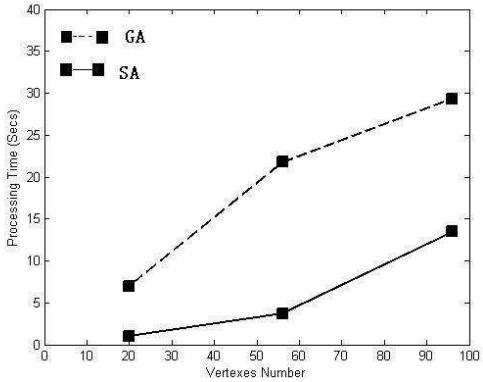

Fig. 8. Comparisons of the computational time between the two methods.

IV. CONCLUSION

A simulated annealing algorithm based method has been proposed to determine the optimal or near-optimal path quickly for a mobile robot in dynamic environments with static and dynamic obstacles. A new algorithm planner has also been developed for enhancement of the efficiency of the path planning algorithm. Simulation results have demonstrated the effectiveness and efficiency of the proposed approach.

ACKNOWLEDGMENT

The authors would like to thank Australia Government's Department of Education, Science and Training (DEST) for its support under the International Science Linkages (ISL) Grant Scheme (Grant Number: CH070083).

REFERENCES

[1] H. Mitchell, "An algorithmic approach to some problems in terrain navigation," *Artificial Intelligence*, vol. 37, pp. 171–201, 1988.
[2] C. O'Dunlaing and C.-K. Yap, "A retraction method for planning the motion of a disc," *Journal of Algorithms*, vol. 6, pp. 104–111, 1982.
[3] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey, "Grid-based mapping for autonomous mobile robot," *Robotics & Autonomous Systems*, vol. 11, pp. 13–21, 1993.
[4] X. Hu and C. Xie, "Niche genetic algorithm for robot path planning," in *Proc. Third Int. Conf. on Natural Computation*, vol. 2. IEEE, 2007, pp. 600–605.
[5] Y. Davidor, *Genetic algorithms and robotics: a heuristic strategy for optimization (Series in Robotics and Automated Systems Vol. 1)*. Singapore: World Scientific Publishing, 1991, pp. 220–225.
[6] A. C. Nearchou, "Path planning of a mobile robot using genetic heuristics," *Robotica*, vol. 16, pp. 575–588, 1998.
[7] Z. Cai and Z. Peng, "The application of a novel encoding mechanism in path planning for a mobile robot," *Robot*, vol. 23, pp. 230–233, 1997.
[8] J. Ayers, "Underwater walking," *Arthropod Structure & Development*, vol. 33, pp. 347–360, 2004.
[9] B. Williams and I. Mahon, "Design of an unmanned underwater vehicle for reef surveying," in *Proc. IFAC Third Symp. on Mechatronic Systems*. Manly NSW, Australia: IFAC, September 2004.
[10] N. Lv and Z. Feng, "Numerical potential field and ant colony optimization based path planning in dynamic environment," in *Proc. Sixth World Congress on Intelligent Control & Automation (WCICA'2006)*, vol. 2. IEEE, 2006, pp. 8966–8970.
[11] X. Xu, K. Xie, and K. Xie, "Path planning and obstacle-avoidance for soccer robot based on artificial potential field and genetic algorithm," in *Proc. Sixth World Congress on Intelligent Control & Automation (WCICA'2006)*, vol. 1. Dalian, China: IEEE, June 2006, pp. 3494 – 3498.
[12] Y. Wang, P. W. Sillitoe, , and J. Mulvaney, "Mobile robot path planning in dynamic environments," *Robotics & Automation*, pp. 71–76, 2007.
[13] Q. Cao, Y. Huan, and J. Zhou, "An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot," in *Proc. Int. Conf. on Intelligent Robots & Systems*, 2006, pp. 3331–3336.
[14] P. Zhang and T. Lv, "Soccer robot path planning based on artificial potential field approach with simulated annealing," *Robotica*, vol. 22, pp. 563–566, 2004.
[15] Wikipedia, "Introduction of simulated annealing algorithm," http://en.wikipedia.org/wiki/Simulated_annealing, The free encyclopedia, retrieved on August 11, 2008.
[16] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Proc. Sixth Int. Conf. on Intelligent System Design & Applications*, vol. 2. IEEE, 2006, pp. 622–627.
[17] Y. Wang, D. Mulvaney, and I. Sillitoe, "Genetic-based mobile robot path planning using vertex heuristics," in *Proc. Conf. on Cybernetics & Intelligent Systems*. IEEE, 2006, pp. 1–6.
[18] Mathworks, "Matlab product family," http://www.mathworks.com/, retrieved on August 10, 2008.